**Security Documentation**

**Introduction**

The security of this application is managed by Spring Security, a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications.

**Security Configuration**

The security configuration is defined in the `SecurityConfig` class (not shown in the provided code). This class extends `WebSecurityConfigurerAdapter` and overrides methods to customize the security settings. The configuration includes defining which URL paths should be secured, setting up the login process, and configuring how to handle exceptions.

**User Authentication**

User authentication is handled by the `UserDetailServiceImpl` class, which implements the `UserDetailsService` interface from Spring Security. This service is used to retrieve user details during the authentication process.

The `loadUserByUsername` method is overridden to load user details based on the username (in this case, the email). If a user with the provided email can't be found, a `UsernameNotFoundException` is thrown.

The user details are returned as a `UserDetails` object, which in this case is a `User` object from Spring Security. This object includes the username, password, and authorities of the user. The authorities are granted roles that can be used for authorization checks.

**Password Encoding**

Password encoding is typically handled by a `PasswordEncoder` (not shown in the provided code). Spring Security provides several `PasswordEncoder` implementations, such as `BCryptPasswordEncoder`. The password encoder is used to encode the password when a user is created and to verify the password during the authentication process.

**Authorization**

Authorization is typically handled by defining access-control rules in the `SecurityConfig` class. These rules can be based on the roles of the authenticated user. For example, you can restrict access to certain URL paths to users with a specific role.

**JWT Authentication**

The application uses JSON Web Token (JWT) for authentication. The `JWTAuthenticationFilter` and `JWTAuthorizationFilter` classes (not shown in the provided code) are used to implement JWT authentication. The `JWTAuthenticationFilter` processes authentication requests and generates a JWT if the authentication is successful. The `JWTAuthorizationFilter` checks the JWT in the request header and sets the authentication in the security context.

**Conclusion**

This is a brief overview of the security part of the application. For more detailed information, please refer to the Spring Security and JWT documentation, as well as the source code of the application.